PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DO SOFTWARE “STICKERZ”: UM APLICATIVO PARA TROCA DE FIGURINHAS DA COPA DO MUNDO

GOIÂNIA
2025

HAYANN GONÇALVES SILVA

DESENVOLVIMENTO DO SOFTWARE “STICKERZ”: UM APLICATIVO PARA TROCA DE FIGURINHAS DA COPA DO MUNDO

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação.

Orientador(a):
Profa. Msc. Lucilia Gomes Ribeiro

GOIÂNIA
2025

HAYANN GONÇALVES SILVA

DESENVOLVIMENTO DO SOFTWARE “STICKERZ”: UM APLICATIVO PARA TROCA DE FIGURINHAS DA COPA DO MUNDO

Trabalho de conclusão de Curso aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Ciência da Computação, em _____/_____/_____.

_____

Orientador(a): Profa. Msc. Lucilia Gomes Ribeiro

_____

Prof. Msc. Fernando Gonçalves Abadia

_____

Prof. Msc. Max Gontijo de Oliveira

GOIÂNIA

2025

RESUMO

Este trabalho apresenta o desenvolvimento do aplicativo móvel "Stickerz", uma solução tecnológica para facilitar a troca de figurinhas da Copa do Mundo entre colecionadores. O projeto utiliza tecnologias modernas como Kotlin para desenvolvimento Android e Spring Boot para o backend, implementando arquitetura REST. O aplicativo incorpora funcionalidades de geolocalização para conectar colecionadores próximos geograficamente que possuem figurinhas compatíveis para troca, sistema de autenticação seguro baseado em JWT e um gerenciamento completo de coleções organizadas por times e países. A metodologia adotada seguiu uma abordagem exploratória e experimental, com desenvolvimento incremental baseado em práticas ágeis, estruturada em sete etapas sequenciais desde análise do problema até validação das funcionalidades. O sistema implementa a arquitetura MVVM para o frontend mobile e MVC com camada de serviços para o backend, garantindo separação adequada de responsabilidades e facilidade de manutenção. O aplicativo oferece interfaces intuitivas para login, cadastro, gerenciamento de coleção, visualização de matches, busca de colecionadores e perfil do usuário. Os resultados demonstram uma solução funcional que otimiza o processo de completar álbuns de figurinhas, reduzindo custos através de trocas inteligentes baseadas em proximidade geográfica e compatibilidade de coleções criando e conectando uma comunidade de colecionadores.

Palavras-chave: Desenvolvimento móvel. Kotlin. Java. Spring. Android. Copa do Mundo. Coleção.

ABSTRACT

This work presents the development of the "Stickerz" mobile application, a technological solution to facilitate World Cup sticker trading among collectors. The project utilizes modern technologies such as Kotlin for Android development and Spring Boot for the backend, implementing REST architecture. The application incorporates geolocation functionalities to connect geographically nearby collectors who have compatible stickers for trading, secure JWT-based authentication system, and complete collection management organized by teams and countries. The adopted methodology followed an exploratory and experimental approach, with incremental development based on agile practices, structured in seven sequential stages from problem analysis to functionality validation. The system implements MVVM architecture for the mobile frontend and MVC with service layer for the backend, ensuring proper separation of concerns and ease of maintenance. The application offers intuitive interfaces for login, registration, collection management, match visualization, collector search, and user profile. The results demonstrate a functional solution that optimizes the process of completing sticker albums, reducing costs through intelligent trades based on geographic proximity and collection compatibility, creating and connecting a community of collectors.

Keywords: Mobile development. Kotlin. Java. Spring. Android. World Cup. Collections.

LISTA DE FIGURAS

LISTA DE SIGLAS

API	Application Interface Program
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol
MVVM	Model View Viewmodel
MVC	Model View Controller
JVM	Java Virtual Machine
UI	User Interface
IDE	Integrated Development Environment

SUMÁRIO

1 INTRODUÇÃO

A Copa do Mundo é um dos eventos esportivos mais icônicos e amplamente celebrados em todo o mundo, além da paixão pelo futebol esse evento traz também a tradição de colecionar figurinhas do evento com o objetivo de completar o álbum.

Com o crescimento do uso de dispositivos móveis e a popularização das redes sociais, a comunicação instantânea tornou-se essencial para a vida moderna. Nesse sentido, um aplicativo para troca de figurinhas da Copa do Mundo tem potencial para ganhar lugar no mercado (BARCELOS, 2020).

A troca de figurinhas representa um problema matemático conhecido como "O Problema do Colecionador", se o colecionador fizer a troca de figurinhas repetidas ele consegue completar o álbum gastando menos do que comprar pacotes de figurinhas a depender puramente da sorte (ROSA, 2019).

Há também o interesse por parte dos colecionadores em adquirir figurinhas consideradas raras pagando quantias em dinheiro, como por exemplo a figurinha rara do Neymar que houve ofertas de até 9 mil reais (FIRMINO, 2022).

Para atender a essa demanda, o desenvolvimento de uma *aplicação móvel* para troca de figurinhas da Copa do Mundo tornou-se uma oportunidade de unir o aprendizado de tecnologias aplicadas à prática com a possibilidade de criar algo novo. A linguagem de programação Kotlin e o framework Spring Boot são tecnologias que oferecem recursos para o desenvolvimento de uma aplicação móvel e um *backend* de alta qualidade e escaláveis (SPRING, 2023).

Uma Interface de Programação de Aplicativos (API) do tipo Transferência de Estado Representacional (REST) é uma aplicação que disponibiliza métodos HTTP que transferem um estado representacional de um objeto de uma aplicação para outra. As APIs RESTful são a base de toda a internet web.

> Uma API REST (também conhecida como API RESTful) é uma Interface de Programação de Aplicativos (API ou API web) que está em conformidade com as limitações do estilo arquitetônico REST e permite a interação com serviços web RESTful. REST significa Representational State Transfer e foi criado pelo cientista da computação Roy Fielding. (REDHAT, 2020).

Kotlin é uma linguagem de programação moderna, que traz muitas melhorias em relação ao Java, como sintaxe mais simples e segura, além de melhorias na performance. Spring Boot é um framework que facilita a criação de aplicações web, oferecendo muitos recursos para o desenvolvimento de APIs, como segurança usando Spring Security, camada de persistência usando Spring Data JPA e Spring MVC. (DEVELOPER.ANDROID, 2023)

Desenvolver uma aplicação móvel em Kotlin e Spring Boot, seguindo os princípios da arquitetura *Model-View-ViewModel* (MVVM) e utilizando boas práticas de desenvolvimento envolve o estudo de documentações de todas as tecnologias. Pode-se utilizar recursos como WebSockets e RabbitMQ para garantir a comunicação assíncrona entre os usuários do Stickerz.

MVVM é um padrão de arquitetura de software amplamente utilizado no desenvolvimento de aplicativos, especialmente em ambientes que suportam interfaces de usuário (UI) como aplicativos para desktop, web e móveis. O MVVM é uma variação do padrão de design Model-View-Controller (MVC) e tem como objetivo separar as preocupações no desenvolvimento de software, facilitando a manutenção e a escalabilidade do código (MICROSOFT, 2023).

> *Spring facilita a criação de aplicativos corporativos Java. Ele fornece tudo que você precisa para adotar a linguagem Java em um ambiente corporativo, com suporte para Groovy e Kotlin como linguagens alternativas na JVM e com flexibilidade para criar vários tipos de arquiteturas dependendo das necessidades de um aplicativo. (SPRING, 2023).*

Segundo a página referência da fundação Mozilla (2023) “WebSockets é uma tecnologia avançada que torna possível abrir uma sessão de comunicação interativa entre o navegador do usuário e um servidor. Com esta API, você pode enviar mensagens para um servidor e receber respostas orientadas a eventos sem ter que consultar o servidor para obter uma resposta.”

1.1 Objetivo Geral

O objetivo geral deste trabalho é desenvolver um aplicativo móvel que otimize o processo de troca de figurinhas da Copa do Mundo entre colecionadores, possibilitando

encontrar colecionadores compatíveis dentro de um raio de distância, utilizando tecnologias modernas de desenvolvimento *mobile*.

1.2 Objetivos Específicos

1. Criar um sistema de perfis de usuário que permita aos colecionadores listar suas figurinhas disponíveis para troca e especificar as figurinhas que estão buscando para completar seus álbuns;
2. **Implementar um sistema de geolocalização** que identifica automaticamente colecionadores próximos geograficamente com potencial para trocas mutuamente benéficas;
3. **Criar uma interface de usuário intuitiva e acessível** que permita o gerenciamento completo da coleção, visualização de oportunidades de troca e comunicação direta entre colecionadores;
4. **Implementar um sistema de autenticação e segurança robusto** utilizando tecnologias modernas como JWT (JSON Web Tokens) e Spring Security para garantir a proteção dos dados dos usuários;
5. **Aplicar arquiteturas de software** como MVVM no frontend mobile e MVC com camada de serviços no backend;
6. **Validar a eficácia da solução** através de testes funcionais, demonstrando que a ferramenta realmente cumpre com o objetivo de conectar colecionadores.

1.3 Justificativa e Relevância

Com o crescimento do uso de dispositivos móveis e a popularização das redes sociais, a comunicação instantânea e a formação de comunidades digitais tornaram-se essenciais para a vida moderna. Nesse contexto, um aplicativo para troca de figurinhas da Copa do Mundo tem potencial significativo para ganhar espaço no mercado digital.

A relevância deste projeto estende-se além do aspecto comercial. Do ponto de vista acadêmico, o desenvolvimento do Stickerz permite a aplicação prática de diversos conceitos fundamentais da Ciência da Computação, incluindo:

- Estruturas de dados e algoritmos na implementação dos sistemas de busca e compatibilidade;

- Engenharia de software na aplicação de padrões arquiteturais e metodologias de desenvolvimento;
- Programação orientada a objetos no desenvolvimento tanto do backend quanto do frontend;
- Bancos de dados na modelagem e implementação da persistência de dados;
- Segurança da informação na implementação de sistemas de autenticação;
- Desenvolvimento mobile utilizando tecnologias modernas como Kotlin e Android SDK.

Para atender a essa demanda, o desenvolvimento de uma aplicação móvel para troca de figurinhas da Copa do Mundo tornou-se uma oportunidade de unir o aprendizado de tecnologias aplicadas à prática com a possibilidade de criar uma solução inovadora que beneficia uma comunidade real de usuários.

1.4 Organização do Trabalho

Este trabalho está estruturado em seis capítulos organizados de forma sequencial e lógica para apresentar o desenvolvimento completo do aplicativo Stickerz.

O Capítulo 1 apresenta a introdução ao tema, contextualizando a importância da troca de figurinhas da Copa do Mundo e a oportunidade de desenvolvimento de uma solução tecnológica. São definidos os objetivos geral e específicos, bem como a justificativa e relevância do projeto para o contexto acadêmico e tecnológico.

O Capítulo 2 contém a fundamentação teórica necessária para compreensão do projeto. Aborda o Problema do Colecionador de Cupons como base, as tecnologias Android e suas arquiteturas, os padrões arquiteturais MVC e MVVM, e as tecnologias utilizadas no desenvolvimento: Java, Kotlin, Spring Framework e MySQL.

O Capítulo 3 descreve os materiais e procedimentos metodológicos adotados. Apresenta a metodologia de pesquisa exploratória e experimental e as ferramentas utilizadas no projeto: IntelliJ IDEA, Android Studio, DBeaver e Postman.

O Capítulo 4 detalha o desenvolvimento do aplicativo, incluindo a descrição geral do produto, o levantamento de requisitos funcionais e não-funcionais, e a modelagem conceitual e relacional do banco de dados.

O Capítulo 5 apresenta os resultados obtidos através das interfaces desenvolvidas, demonstrando as funcionalidades implementadas nas telas de login, cadastro, busca de colecionadores, matchs, álbum da copa e perfil do usuário.

O Capítulo 6 apresenta as considerações finais, destacando os objetivos alcançados, as limitações identificadas e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Problema do Colecionador de Cupons

O Problema do Colecionador de Figurinhas é uma formulação clássica da Teoria das Probabilidades que busca determinar o número esperado de tentativas necessárias para completar uma coleção composta por *n* itens distintos, quando cada item é obtido aleatoriamente através de um novo pacote adquirido (ROSA, 2019).

No caso específico das figurinhas, a questão central é determinar quantas compras são necessárias até que todas as figurinhas distintas de um álbum sejam obtidas. A dificuldade aumenta conforme o número de figurinhas restantes diminui, já que, com o avanço da coleção, a probabilidade de encontrar uma nova figurinha se torna cada vez menor.

No início da coleção, é relativamente fácil conseguir figurinhas novas, pois quase todas ainda estão faltando. No entanto, à medida que a coleção se aproxima da completude, os colecionadores passam a encontrar com mais frequência figurinhas repetidas. Essa situação gera um aumento gradual da dificuldade e dos custos envolvidos, o que torna a reta final da coleção especialmente demorada e onerosa.

Como o objetivo deste capítulo não é aprofundar na matemática deste problema e sim em como aplicar essa ideia no aplicativo Stickerz, fica definido alguns critérios que deverão ser abrangidos para que o aplicativo cumpra seu propósito, esses critérios serão apresentados no capítulo 4 sobre o aplicativo:

2.2 Android

O Android é um sistema operacional móvel desenvolvido pelo Google, baseado no kernel Linux e projetado primariamente para dispositivos touchscreen como smartphones e tablets. Lançado oficialmente em 2008, o Android rapidamente se estabeleceu como a plataforma móvel dominante globalmente, sendo adotado por diversos fabricantes de hardware como Samsung, Xiaomi, Motorola e OnePlus (SHEIKH, 2013).

2.2.1 Arquitetura e Componentes

A arquitetura do Android é organizada em camadas, seguindo um modelo em pilha que proporciona um ambiente robusto para o desenvolvimento de aplicativos. Segundo Meier (2012), as principais camadas da arquitetura Android incluem:

- **Kernel Linux**: Forma a fundação do sistema operacional, fornecendo serviços essenciais como gerenciamento de memória, segurança e drivers de hardware.
- **Camada de Abstração de Hardware (HAL)**: Fornece interfaces padronizadas que expõem as capacidades do hardware do dispositivo para a camada superior.
- **Runtime do Android**: Inclui o Android Runtime (ART), que substituiu a Dalvik Virtual Machine a partir do Android 5.0, e as bibliotecas de núcleo.
- **Bibliotecas Nativas**: Conjunto de bibliotecas C/C++ utilizadas por vários componentes do sistema.
- **Framework de Aplicação**: Fornece as APIs que formam os blocos de construção para o desenvolvimento de aplicativos.
- **Aplicações**: Camada mais alta contendo os aplicativos pré-instalados e os instalados pelo usuário.

Os aplicativos Android são compostos por quatro tipos principais de componentes, cada um com um propósito específico e ciclo de vida distinto (GARGENTA, 2011):

- **Activities**: Representam uma única tela com interface de usuário, gerenciando a interação do usuário com a informação visual.
- **Services**: Componentes que executam operações de longa duração em segundo plano, sem fornecer uma interface de usuário.
- **Content Providers**: Gerenciam o acesso a dados estruturados, encapsulando os dados e fornecendo mecanismos para a segurança dos dados.
- **Broadcast Receivers**: Respondem a anúncios de broadcast do sistema ou de aplicativos.

2.2.2 Desenvolvimento para Android

O desenvolvimento para Android é tradicionalmente realizado utilizando Java ou Kotlin (linguagem oficialmente suportada desde 2017), em conjunto com o Android SDK

(Software Development Kit) e o ambiente de desenvolvimento integrado Android Studio, que substituiu o Eclipse como IDE oficial em 2014 (DEITEL et al., 2013).

Kotlin, uma linguagem estaticamente tipada desenvolvida pela JetBrains, tem ganhado significativa adoção devido à sua interoperabilidade com Java e recursos que promovem maior segurança e produtividade no desenvolvimento. Desde 2019, o Google passou a priorizar Kotlin para o desenvolvimento Android, refletindo sua crescente importância no ecossistema (KOTLINLANG, 2024).

O processo de desenvolvimento segue um fluxo estruturado que inclui (DEITEL et al., 2016):

- **Configuração do ambiente**: Instalação do Android Studio e SDKs necessários.
- **Criação do projeto**: Definição da estrutura básica do aplicativo, configurações de compatibilidade e dependências.
- **Desenvolvimento da interface**: Utilizando XML para layouts.
- **Implementação da lógica**: Codificação das funcionalidades utilizando Java ou Kotlin.
- **Testes**: Utilização de frameworks como JUnit, Espresso para testes de UI e Mockito para testes de unidade.
- **Distribuição**: Geração do Android Package (APK) ou Bundle Android App e publicação na Google Play Store.

2.3 Arquiteturas para desenvolvimento de aplicativos móveis

A arquitetura de software adotada influencia diretamente a manutenibilidade, escalabilidade e desempenho dos aplicativos móveis (Martin, 2018). Este capítulo apresenta as arquiteturas que foram utilizadas no desenvolvimento da aplicação Stickerz.

2.3.1 Model-View-Controller (MVC)

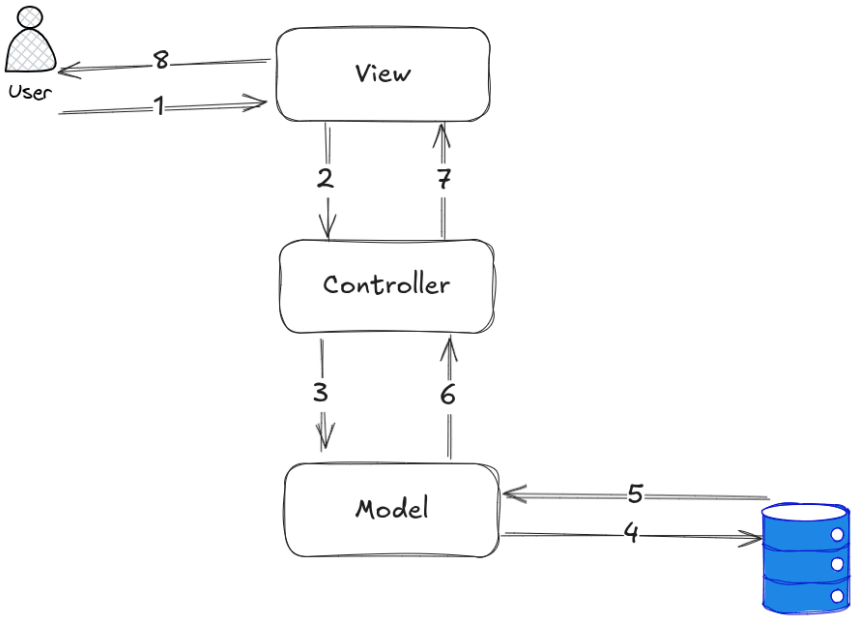
O padrão MVC divide a aplicação em três componentes principais (GeeksforGeeks, 2020):

- ***Model***: Representa os dados e a lógica de negócios

- ***View***: Responsável pela apresentação e interface do usuário
- ***Controller***: Processa as entradas do usuário e coordena a interação entre Model e View

Este padrão é o mais frequentemente utilizado no desenvolvimento mobile, correspondendo a 41% das aplicações segundo pesquisa recente (Daily.dev, 2023), porém apresenta limitações em aplicações complexas, onde o Controller frequentemente acumula muitas responsabilidades.

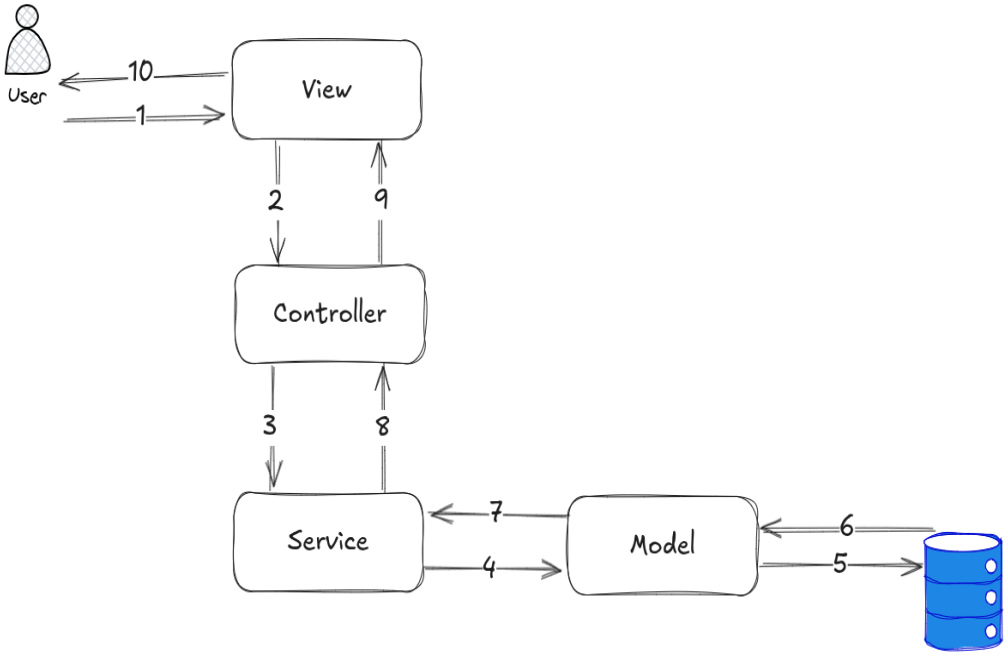
Figura 1 - Representação do modelo MVC.



Fonte: Autoria própria.

O padrão MVC é comumente usado adicionando uma quarta camada, a do *Service*, para separar a lógica de negócio da camada de *Controller* e da camada de Modelo.

Figura 2 - Representação do modelo MVC com a camada Service.



Fonte: Autoria própria.

O padrão MVC com a camada do Service foi o padrão utilizado no desenvolvimento *backend* da aplicação Stickerz.

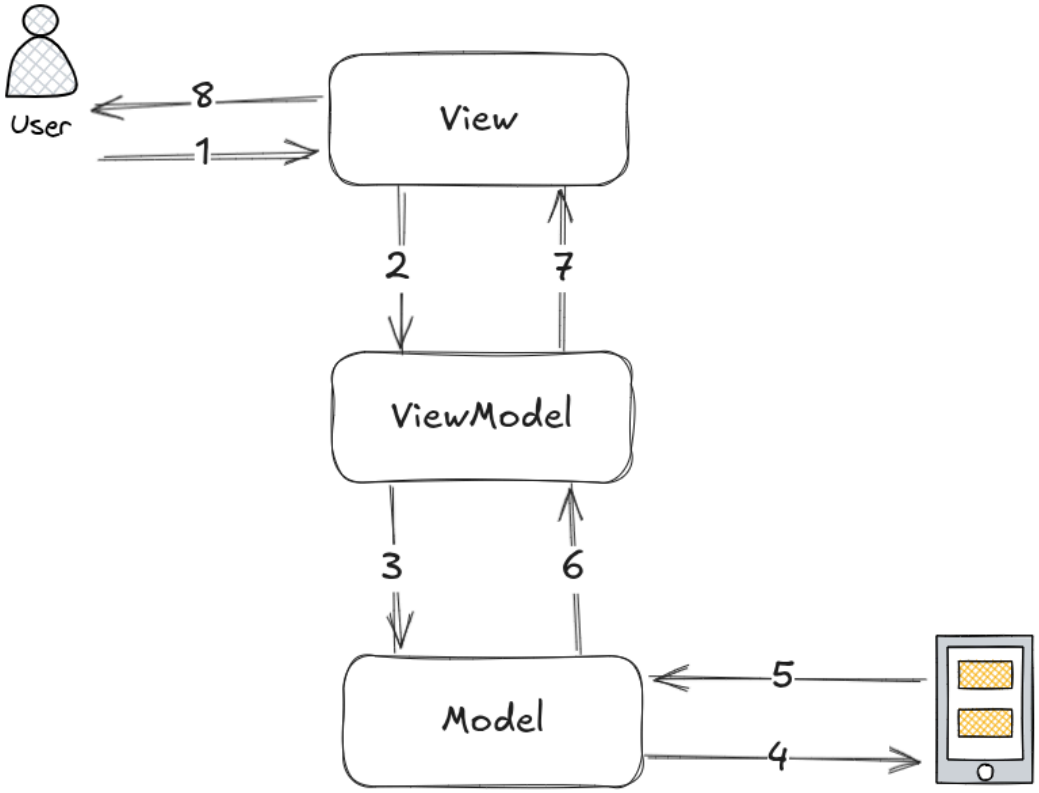
2.3.2 Model-View-ViewModel (MVVM)

O padrão MVVM permite uma separação mais clara entre UI e lógica de negócios (GeeksforGeeks, 2025):

- **Model**: Responsável pelos dados e regras de negócios
- **View**: Representa a interface com o usuário
- **ViewModel**: Expõe dados do Model para a View e gerencia o estado da UI

O MVVM tornou-se popular em plataformas que suportam data binding, como Android com Jetpack e iOS com SwiftUI (Geeksforgeeks, 2025). Esta arquitetura permite uma atualização automática da UI quando os dados mudam, facilitando o desenvolvimento e manutenção.

Figura 3 - Representação do modelo MVVM.



Fonte: Autoria própria.

O padrão MVVM foi utilizado no desenvolvimento da aplicação móvel Android do *software* Stickerz.

2.4 Java

Java é uma linguagem de programação de alto nível, orientada a objetos e multiplataforma, utilizada para o desenvolvimento de uma variedade de aplicativos, desde aplicações web e móveis até sistemas empresariais complexos. Java foi projetada com o objetivo de ser uma linguagem simples, robusta, portável e segura. Criada pela Sun Microsystems e lançada em 1995, sob o slogan ”Write once, run anywhere”, que traduzindo fica “Escreva uma vez, execute em qualquer lugar” (BRITANNICA, 2025).

Uma das principais características do Java é a sua portabilidade, graças à JVM (Java Virtual Machine), que permite que o código Java seja executado em qualquer plataforma que tenha uma implementação da JVM disponível. Isso faz com que Java não dependa de um sistema operacional e a torna ideal para o desenvolvimento de aplicativos que precisam ser executados em diferentes sistemas operacionais.

Além disso, Java possui uma sintaxe limpa e fácil de aprender, o que a torna acessível para desenvolvedores de diferentes níveis de experiência. Ela suporta programação orientada a objetos, encapsulamento, herança e polimorfismo, proporcionando uma estrutura sólida para o desenvolvimento de software modular e reutilizável.

2.5 Kotlin

Kotlin é uma linguagem de programação moderna, que traz muitas melhorias em relação ao Java, como sintaxe mais simples e segura, além de melhorias na performance. Spring Boot é um framework que facilita a criação de aplicações web, oferecendo muitos recursos para o desenvolvimento de APIs, como segurança usando Spring Security, camada de persistência usando Spring Data JPA e Spring MVC (KOTLINLANG, 2024).

2.6 Spring Framework

O Spring Framework é um dos frameworks mais populares para desenvolvimento de aplicativos Java. Ele fornece um conjunto abrangente de recursos para facilitar o desenvolvimento de aplicativos empresariais robustos e escaláveis. Uma das características mais marcantes do Spring é a inversão de controle (IoC), que permite que os objetos sejam injetados em outros objetos pelo container do Spring, facilitando a configuração e a manutenção do código.

Além disso, o Spring oferece módulos para diversas áreas, como persistência de dados (Spring Data), desenvolvimento web (Spring MVC), segurança (Spring Security) e integração com outras tecnologias, como JPA, Hibernate e RESTful APIs. Sua modularidade e extensibilidade tornam o Spring uma escolha popular para uma ampla gama de aplicações Java, desde pequenos projetos até sistemas corporativos complexos.

2.6.1 Spring Boot

O Spring Boot é um projeto do ecossistema Spring que visa simplificar radicalmente o processo de criação e implantação de aplicativos Java. Ele oferece uma abordagem convention over configuration, o que significa que muitas configurações são automaticamente assumidas com base nas convenções padrão, reduzindo assim a quantidade de configuração manual necessária.

Uma das características mais marcantes do Spring Boot é o seu mecanismo de inicialização automática, que configura automaticamente o aplicativo com base nas dependências presentes no classpath. Isso significa que, com o Spring Boot, você pode criar rapidamente um aplicativo funcional com apenas algumas linhas de código, permitindo que os desenvolvedores se concentrem no desenvolvimento de recursos de negócios em vez de configurações de infraestrutura.

2.6.2 Spring Web MVC

O Spring MVC é um dos módulos mais importantes do Spring Framework, projetado para facilitar o desenvolvimento de aplicativos web baseados em Java. Ele segue o padrão arquitetural MVC, onde o modelo representa os dados da aplicação, a visualização é responsável pela apresentação dos dados ao usuário e o controlador manipula as requisições do usuário, processando e respondendo a elas.

O Spring MVC oferece uma estrutura flexível para desenvolver aplicativos web escaláveis e de fácil manutenção. Ele fornece recursos como mapeamento de URLs para métodos de controle, suporte a anotações para simplificar a configuração, resolução de exceções e validação de dados integrada.

Além disso, o Spring MVC é altamente extensível e pode ser facilmente integrado com outros módulos do Spring, como Spring Security para autenticação e autorização, Spring Data para acesso a banco de dados e Spring Boot para configuração simplificada e inicialização rápida de aplicativos.

2.6.3 Spring Data

O Spring Data é um projeto dentro do ecossistema Spring que simplifica o acesso e a manipulação de dados em aplicativos Java, fornecendo uma abstração de alto nível sobre as tecnologias de acesso a dados. Ele oferece suporte para uma ampla gama de tecnologias de persistência de dados, como bancos de dados relacionais, bancos de dados NoSQL como MongoDB e Cassandra, e tecnologias de busca como Elasticsearch.

Uma das características mais poderosas do Spring Data é a sua abordagem baseada em repositórios, que permite aos desenvolvedores escrever consultas de dados de forma declarativa, sem a necessidade de escrever código boilerplate para acessar o banco de dados.

Além disso, o Spring Data oferece recursos avançados como paginação, ordenação, consulta personalizada, audição de alterações e integração com outras partes do ecossistema Spring, como Spring Boot e Spring MVC. Ele também fornece suporte para transações declarativas e controle de transação baseado em anotações, facilitando a implementação de operações de banco de dados seguras e eficientes.

2.6.4 Spring Security

O Spring Security é um projeto do ecossistema Spring que fornece diversos recursos para lidar com os aspectos de segurança em aplicações Java. Ele oferece funcionalidades para autenticação, autorização e proteção da aplicação, fornecendo aos desenvolvedores uma interface de programação pronta para o uso.

Uma das principais características do Spring Security é a sua flexibilidade e extensibilidade. Ele permite aos desenvolvedores integrar facilmente diferentes mecanismos de autenticação, como autenticação baseada em formulários, autenticação baseada em token, autenticação baseada em certificado e autenticação OAuth/OpenID, entre outros.

Além disso, o Spring Security fornece um modelo de autorização robusto que permite aos desenvolvedores definir regras de acesso granulares com base em papéis de usuário, expressões de segurança e outras condições personalizadas.

2.7 MySQL

O MySQL é um sistema de gerenciamento de banco de dados (SGBD) desenvolvido originalmente pela empresa sueca MySQL AB em 1995 por Michael Widenius, David Axmark e Allan Larsson. Baseado na linguagem de consulta estruturada *Structured Query Language* (SQL), o MySQL foi projetado inicialmente como uma solução open-source rápida, confiável e fácil de usar.

O sistema utiliza uma arquitetura cliente-servidor robusta, suportando múltiplos mecanismos de armazenamento intercambiáveis, sendo o InnoDB o padrão atual, que oferece suporte completo a transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), chaves estrangeiras e recuperação de falhas (MYSQL, 2025).

3 MATERIAIS E PROCEDIMENTOS METODOLÓGICOS

O objetivo deste capítulo é expor as metodologias e estratégias adotadas para a realização deste projeto, assim como será feita uma introdução sobre as ferramentas utilizadas na implementação do aplicativo Stickerz para troca de figurinhas entre colecionadores.

3.1 Métodos

Esta pesquisa segundo sua natureza é um resumo de assunto que busca sistematizar uma área de conhecimento relacionada ao desenvolvimento de aplicações móveis para, indicando sua evolução histórica e estado da arte (WAZLAWICK, 2014).

Segundo seus objetivos é exploratória pois não há necessariamente uma hipótese específica, a ideia é desenvolver um aplicativo móvel para troca de figurinhas entre colecionadores e iremos explorar as possibilidades de construção para chegarmos a esse resultado (WAZLAWICK, 2014).

Segundo seus procedimentos técnicos, esta pesquisa é bibliográfica, documental e experimental (WAZLAWICK, 2014). Segundo Gil (2017), uma pesquisa bibliográfica é elaborada com base em material já publicado como materiais impressos, livros, revistas, jornais, teses, dissertações e anais de eventos científicos. Neste caso, fundamenta-se em literatura sobre desenvolvimento de aplicações móveis e tecnologias Spring Boot e Kotlin.

A pesquisa documental complementa a bibliográfica através da análise de documentações técnicas das ferramentas utilizadas, manuais de APIs e especificações de frameworks. Já o caráter experimental manifesta-se na implementação prática do aplicativo, onde são testadas diferentes abordagens técnicas para validar a eficácia da solução proposta.
Adotou-se o método de desenvolvimento incremental e iterativo, baseado em práticas ágeis, permitindo ajustes e melhorias contínuas durante o processo de implementação. Essa abordagem possibilita a validação constante das funcionalidades desenvolvidas e garante maior alinhamento com os requisitos identificados.

Para executar o projeto, adotou-se uma metodologia estruturada em sete etapas sequenciais:

1. **Análise do problema**: Entender como um aplicativo para troca de figurinhas pode facilitar o encontro de colecionadores.

2. **Levantamento e especificação dos requisitos funcionais e não funcionais**: Identificação detalhada das necessidades dos usuários e definição das funcionalidades essenciais do sistema.
3. **Pesquisa bibliográfica sobre tecnologias e ferramentas**: Estudo aprofundado das tecnologias framework Spring, Kotlin e autenticação JWT aplicadas ao desenvolvimento de aplicações móveis.
4. **Projeto da arquitetura do sistema**: Definição da arquitetura REST, modelagem do banco de dados, estruturação dos microsserviços e planejamento da comunicação entre os componentes da solução.
5. **Desenvolvimento do protótipo e implementação**: Codificação das funcionalidades de autenticação, gerenciamento de coleções, sistema de localização e algoritmo de compatibilidade entre colecionadores.
6. **Execução de testes e validação**: Realização de testes unitários, testes de integração, testes de usabilidade e validação das funcionalidades implementadas.

3.2 Materiais

3.2.1 Intellij Idea

O IntelliJ IDEA é um ambiente de desenvolvimento integrado (IDE) escrito em Java para desenvolvimento de software em Java, Kotlin, Groovy e outras linguagens baseadas na JVM. É desenvolvido pela JetBrains como um IDE para desenvolvimento profissional em Java e Kotlin, criado para promover a produtividade, garantir código de qualidade e oferecer diversos suportes ao uso das linguagens de programação pelo desenvolvedor (JETBRAINS, 2025).

3.2.2 Android Studio

O Android Studio é uma IDE baseada na versão Community Edition do IntelliJ IDEA da JetBrains, sendo disponibilizado gratuitamente através da Licença Apache 2.0 e estando disponível para as plataformas Microsoft Windows, macOS e GNU/Linux. A IDE oferece aos desenvolvedores um ambiente de desenvolvimento integrado otimizado para aplicações Android, proporcionando uma experiência ainda mais aprimorada. O que distingue o Android Studio de outros ambientes de desenvolvimento é sua especialização profunda no ecossistema

Android. Além do poderoso editor de códigos e das ferramentas para desenvolvedores herdadas do IntelliJ (DEVELOPER.ANDROID, 2025).

3.2.3 Dbeaver

O DBeaver é um cliente de banco de dados universal e gratuito, desenvolvido em Java, é uma ferramenta popular e versátel para administração e desenvolvimento de bancos de dados. Criado inicialmente por Serge Rider em 2010, o DBeaver oferece suporte a diversos sistemas de gerenciamento de banco de dados, incluindo MySQL, PostgreSQL, SQLite, Oracle, Microsoft SQL Server, IBM DB2, Apache Cassandra, MongoDB e muitos outros.

A arquitetura do DBeaver permite tanto operações básicas quanto avançadas de administração de banco de dados, incluindo navegação em estruturas de dados, execução de consultas SQL, edição de dados, geração de diagramas ER (Entidade-Relacionamento), importação e exportação de dados em diversos formatos, e ferramentas de migração entre diferentes sistemas de banco de dados (DBEAVER, 2025).

3.2.4 Postman

O Postman é uma plataforma colaborativa líder para desenvolvimento e teste de APIs, que auxilia os desenvolvedores a interagir e validar serviços web. Originalmente concebido como uma extensão simples para o navegador Chrome em 2012 por Abhinav Asthana, o Postman evoluiu para se tornar uma aplicação desktop robusta e posteriormente uma plataforma completa baseada na nuvem.

A ferramenta permite aos desenvolvedores criar, enviar, testar e documentar requisições HTTP através de uma interface gráfica amigável, eliminando a complexidade de trabalhar diretamente com comandos de linha ou código para testar endpoints de API. Sua funcionalidade principal inclui suporte para todos os métodos HTTP (GET, POST, PUT, DELETE, PATCH, etc.), gerenciamento de autenticação, manipulação de headers personalizados, e capacidade de trabalhar com diferentes formatos de dados como JSON, XML e form-data (POSTMAN, 2025).

4 DESENVOLVIMENTO

4.1 Descrição geral do produto

O Stickerz é um aplicativo móvel desenvolvido especificamente para colecionadores de figurinhas que desejam realizar trocas de forma prática e organizada. O produto utiliza geolocalização para conectar usuários próximos geograficamente que possuem figurinhas compatíveis para troca, criando uma rede local de colecionadores.

O aplicativo permite que os usuários cadastrem e gerenciem completamente sua coleção de figurinhas, organizadas por países e times, com controle preciso das quantidades possuídas e progresso de conclusão do álbum.

A plataforma identifica automaticamente outros colecionadores na região que têm figurinhas necessárias para completar as coleções, apresentando sugestões de troca personalizadas.

A interface é intuitiva e focada na experiência do usuário, oferece funcionalidades como definição de área de busca personalizável, contato direto via telefone, visualização detalhada das trocas propostas e gerenciamento do perfil.

O sistema de autenticação usando Spring Security garante segurança e permite que cada colecionador mantenha sua coleção e preferências sincronizadas.

Dessa forma, o aplicativo Stickerz facilita o encontro de colecionadores com figurinhas repetidas e podem ser trocadas por outras, ajudando a completar o álbum de forma mais rápida e com menor custo.

O aplicativo Stickerz deverá atender 3 premissas fundamentais, que são:

- Colecionador A tem figurinhas **repetidas** que colecionador B **precisa** (faltantes);
- Colecionador B tem figurinhas **repetidas** que colecionador A **precisa** (faltantes);
- Colecionador A e B estão dentro de um raio de distância (ex: 10km).

4.2 Requisitos funcionais

Tabela 1 - Requisitos funcionais.

Identificador	Requisito Funcional	Descrição detalhada
RF01	Autenticação	O sistema deve permitir que o usuário se cadastre fornecendo email, nome, telefone, senha e device ID
RF02	Autenticação	O sistema deve permitir que o usuário faça login utilizando email, senha e device ID
RF03	Autenticação	O sistema deve validar credenciais e gerar tokens de acesso (JWT) e refresh token
RF04	Autenticação	O sistema deve permitir renovação de tokens através do refresh token
RF05	Autenticação	O sistema deve permitir logout do dispositivo atual invalidando a sessão
RF06	Autenticação	O sistema deve permitir logout de todos os dispositivos do usuário
RF07	Gerenciamento de Coleção	O sistema deve permitir que o usuário cadastre sua coleção de figurinhas
RF08	Gerenciamento de Coleção	O sistema deve permitir visualizar a coleção organizada por times
RF09	Gerenciamento de Coleção	O sistema deve permitir definir a quantidade de cada figurinha possuída
RF10	Gerenciamento de Coleção	O sistema deve permitir atualizar as quantidades das figurinhas
RF11	Localização	O sistema deve permitir que o usuário atualize sua localização (latitude, longitude, precisão)
RF12	Localização	O sistema deve permitir definir o range de distância para busca de colecionadores

RF13	Localização	O sistema deve permitir visualizar o range de distância configurado
RF14	Localização	O sistema deve permitir atualizar o range de distância
RF15	Busca de Colecionadores	O sistema deve buscar colecionadores compatíveis baseado na localização e range definido
RF16	Busca de Colecionadores	O sistema deve identificar figurinhas compatíveis para troca entre colecionadores
RF17	Busca de Colecionadores	O sistema deve apresentar colecionadores compatíveis na tela inicial
RF18	Interação	O sistema deve permitir aceitar ou rejeitar propostas de troca
RF19	Segurança	O sistema deve controlar tentativas de login falhadas
RF20	Segurança	O sistema deve validar dispositivos para renovação de tokens

4.3 Requisitos não-funcionais

Tabela 2 - Requisitos não funcionais.

Identificador	**Requisito Não Funcional**	**Descrição Detalhada**
RNF01	Segurança	O sistema deve criptografar senhas utilizando BCrypt
RNF02	Segurança	O sistema deve implementar autenticação baseada em JWT
RNF03	Segurança	O sistema deve validar formato de email
RNF04	Segurança	O sistema deve controlar sessões por dispositivo

RNF05	Performance	O sistema deve registrar tempo de execução das operações
RNF06	Observabilidade	O sistema deve implementar logging estruturado para todas as operações
RNF07	Observabilidade	O sistema deve registrar eventos de autenticação, localização e coleção
RNF08	Arquitetura	O sistema deve seguir arquitetura REST
RNF09	Arquitetura	O sistema deve ser desenvolvido em Spring Boot com Kotlin
RNF10	Persistência	O sistema deve utilizar JPA para persistência de dados
RNF11	Configuração	O sistema deve permitir configuração de CORS
RNF12	Configuração	O sistema deve desabilitar CSRF para API REST
RNF13	Transações	O sistema deve garantir transações atômicas para operações de cadastro
RNF14	Validação	O sistema deve validar dados de entrada em todas as operações
RNF15	Escalabilidade	O sistema deve suportar múltiplas sessões por usuário em diferentes dispositivos
RNF16	Disponibilidade	O sistema deve implementar tratamento de exceções customizadas
RNF17	Usabilidade	O sistema deve retornar mensagens de erro em português
RNF18	Compatibilidade	O sistema deve funcionar com diferentes tipos de dispositivos (via User-Agent)

RNF19	Manutenibilidade	O sistema deve seguir padrões de clean code e separação de responsabilidades

4.4 Modelo Conceitual

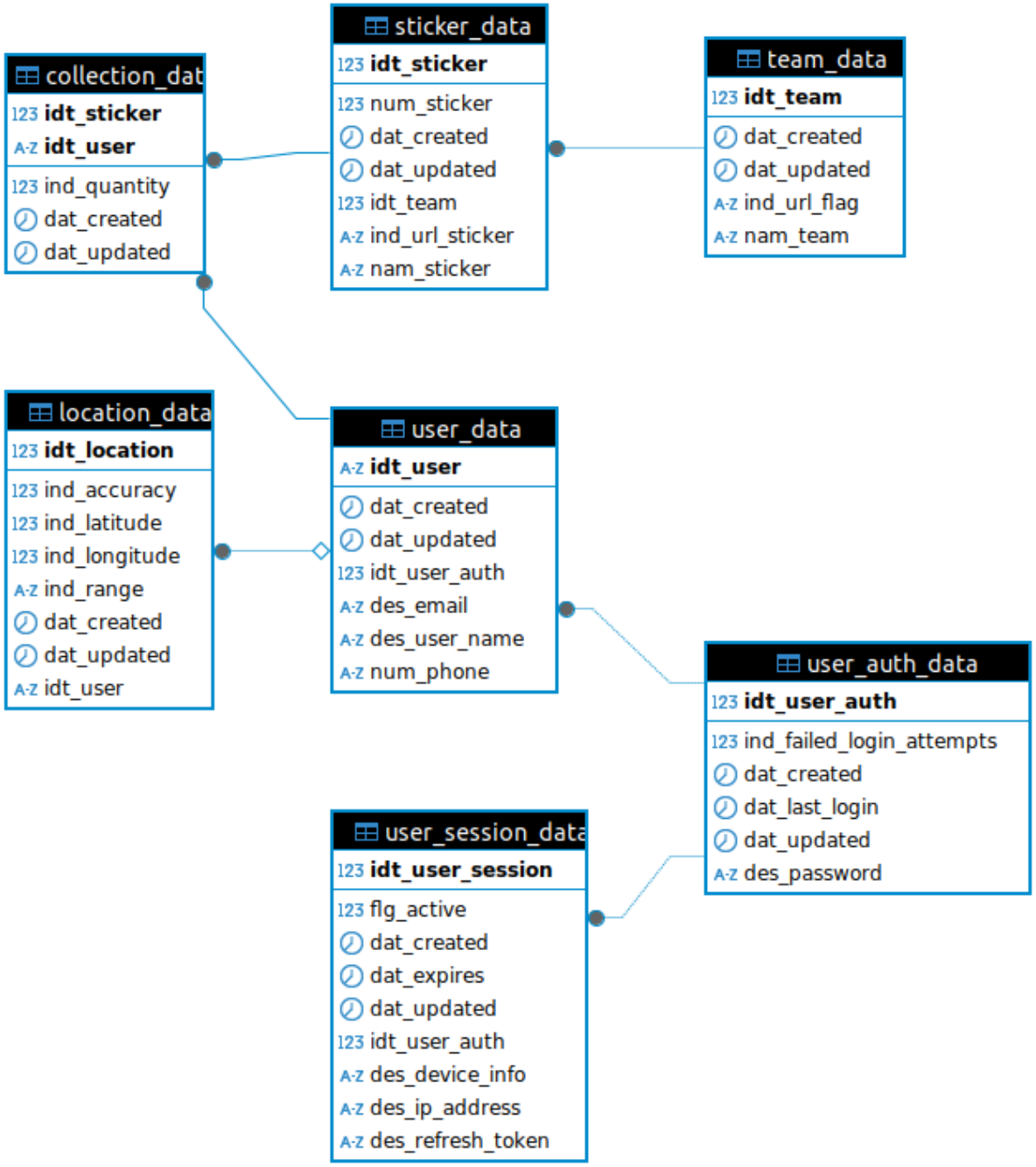
1. **Entidade: USER_AUTH_DATA**
    a. **idt_user_auth**: Identificador único do usuário para autenticação.
    b. des_password: Senha do usuário.
    c. ind_failed_login_attempts: Número de tentativas de login falhas.
    d. dat_created: Data de criação do registro.
    e. dat_last_login: Data do último login.
    f. dat_updated: Data da última atualização.
2. **Entidade: USER_DATA**
    a. **idt_user**: Identificador único do usuário.
    b. idt_user_auth: Identificador do vínculo com dados de autenticação.
    c. des_email: E-mail do usuário.
    d. des_user_name: Nome do usuário.
    e. num_phone: Telefone do usuário.
    dat_created: Data de criação do registro.
    f. dat_updated: Data da última atualização.
3. **Entidade: USER_SESSION_DATA**
    a. **idt_user_session:** Identificador único da sessão.
    b. idt_user_auth: Identificador do vínculo com dados de autenticação.
    c. flg_active: Indicador de sessão ativa.
    d. des_device_info: Informações do dispositivo.
    e. des_ip_address: Endereço IP da sessão.
    f. des_refresh_token: Token de atualização.
    g. dat_created: Data de criação da sessão.
    h. dat_expires: Data de expiração da sessão.
    i. dat_updated: Data da última atualização.
4. **Entidade: TEAM_DATA**
    a. **idt_team**: Identificador único do time.

b. nam_team: Nome do time.
c. ind_url_flag: URL da imagem da bandeira do time.
d. dat_created: Data de criação do registro.
e. dat_updated: Data da última atualização.

5. **Entidade: STICKER_DATA**
    a. **idt_sticker**: Identificador único da figurinha.
    b. idt_team: Identificador do vínculo com o time.
    c. num_sticker: Número da figurinha.
    d. nam_sticker: Nome da figurinha.
    e. ind_url_sticker: URL da imagem da figurinha.
    f. dat_created: Data de criação do registro.
    g. dat_updated: Data da última atualização.
6. **Entidade: LOCATION_DATA**
    a. **idt_location**: Identificador único da localização.
    b. idt_user: Identificador do vínculo com o usuário.
    c. ind_latitude: latitude da localização.
    d. ind_longitude: longitude da localização.
    e. ind_accuracy: precisão da localização.
    f. ind_range: raio de alcance ou área.
    g. dat_created: Data de criação do registro.
    h. dat_updated: Data da última atualização.
7. **Entidade: COLLECTION_DATA**
    a. **idt_sticker**: Identificador do vínculo com a figurinha.
    b. **idt_user**: Identificador do vínculo com o usuário.
    c. ind_quantity: Quantidade de figurinhas do usuário.
    d. dat_created: Data de criação do registro.
    e. dat_updated: Data da última atualização.

4.5 Modelo Relacional

Figura 4 - Modelo relacional.



Fonte: Autoria própria.

4.6 Pré Carga de Tabelas

O sistema implementa uma estratégia de pré-carga de dados essenciais para garantir que as informações básicas sobre times e figurinhas estejam disponíveis no banco de dados desde o primeiro acesso da aplicação. Esta abordagem elimina a necessidade de cadastro manual inicial e proporciona uma experiência mais fluida para os usuários.

Esta pré-carga é executada através de comandos INSERT estruturados que populam a tabela com informações padronizadas e consistentes de todas as seleções participantes da Copa do Mundo.

4.6.1 Tabela TEAM_DATA

A tabela TEAM_DATA receberá uma pré-carga contendo todos os times participantes da Copa do Mundo. Os dados inseridos incluem:

- **Informações básicas dos times**: nome oficial, código FIFA, país de origem;
- **Dados visuais**: cores principais, logo oficial, bandeira nacional;
- **Metadados**: grupo de classificação, ranking FIFA, histórico de participações.

4.6.2 Tabela STICKER_DATA

A tabela STICKER_DATA é pré-carregada com o catálogo completo de figurinhas disponíveis no álbum digital. O processo inclui:

- **Figurinhas de jogadores**: dados individuais de cada atleta convocado, incluindo posição, número da camisa, estatísticas básicas;
- **Associação com times**: cada figurinha é vinculada ao seu respectivo time através de chaves estrangeiras.

5 RESULTADOS ESPERADOS

5.1 Tela de login

Esta é a primeira tela exibida quando o aplicativo Stickerz é iniciado, caso o usuário ainda não esteja autenticado. Sua função é permitir que o usuário acesse sua conta através do e-mail e senha ou crie uma nova. A tela é composta por um formulário centralizado que solicita as credenciais de acesso, oferecendo também a opção de recuperação de senha, como mostra a figura 5.

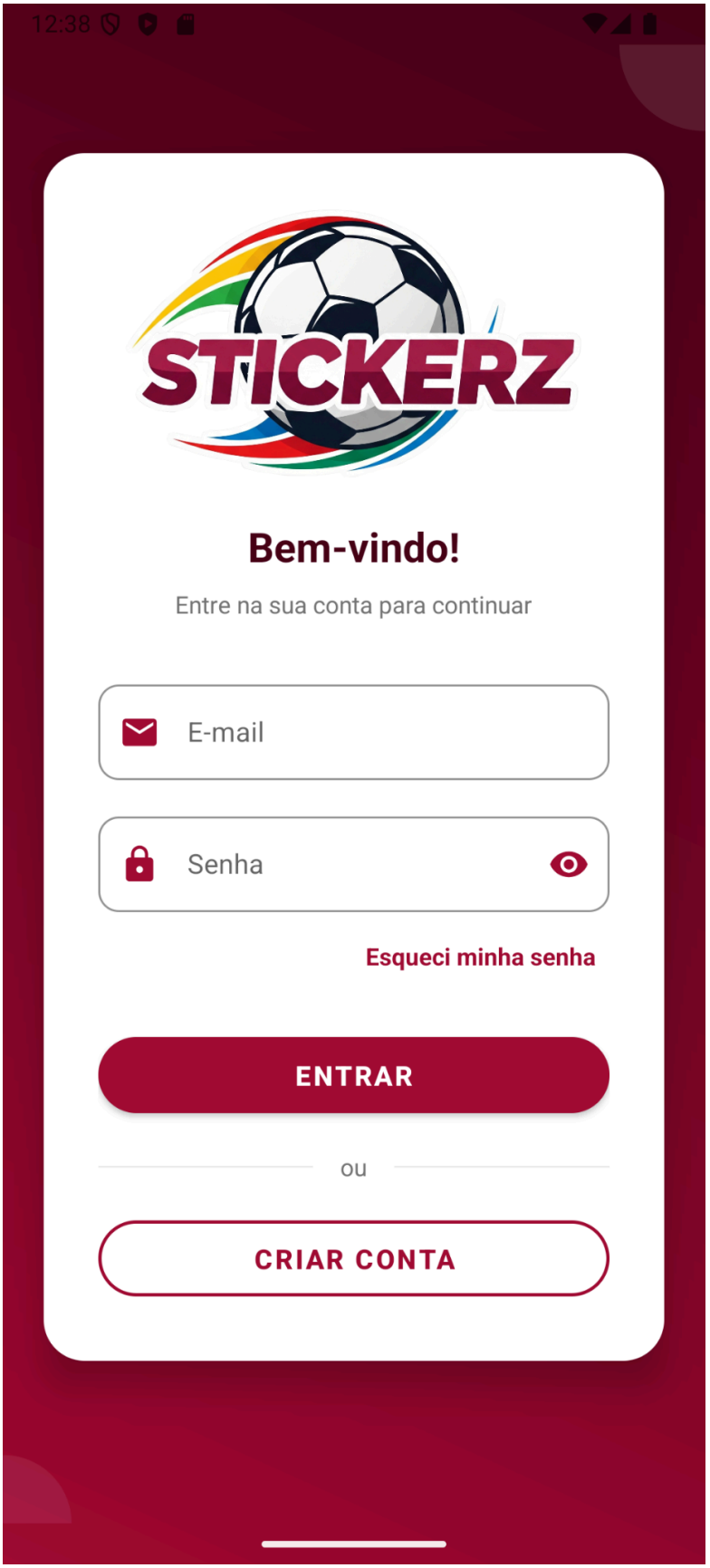
Há um destaque visual com o logotipo da aplicação. Quando o usuário ainda não está autenticado, somente as opções de login e cadastro são disponibilizadas. A navegação para outras funcionalidades só ocorre após a autenticação bem-sucedida.

Caso o usuário tente realizar o login mas por algum motivo falhe, será exibida uma mensagem de erro na cor vermelha embaixo do campo e o campo ficará em destaque, como mostra a figura 6.

Caso o usuário clique na opção de criar conta ele será redirecionado para a tela de cadastro vista na figura 7.

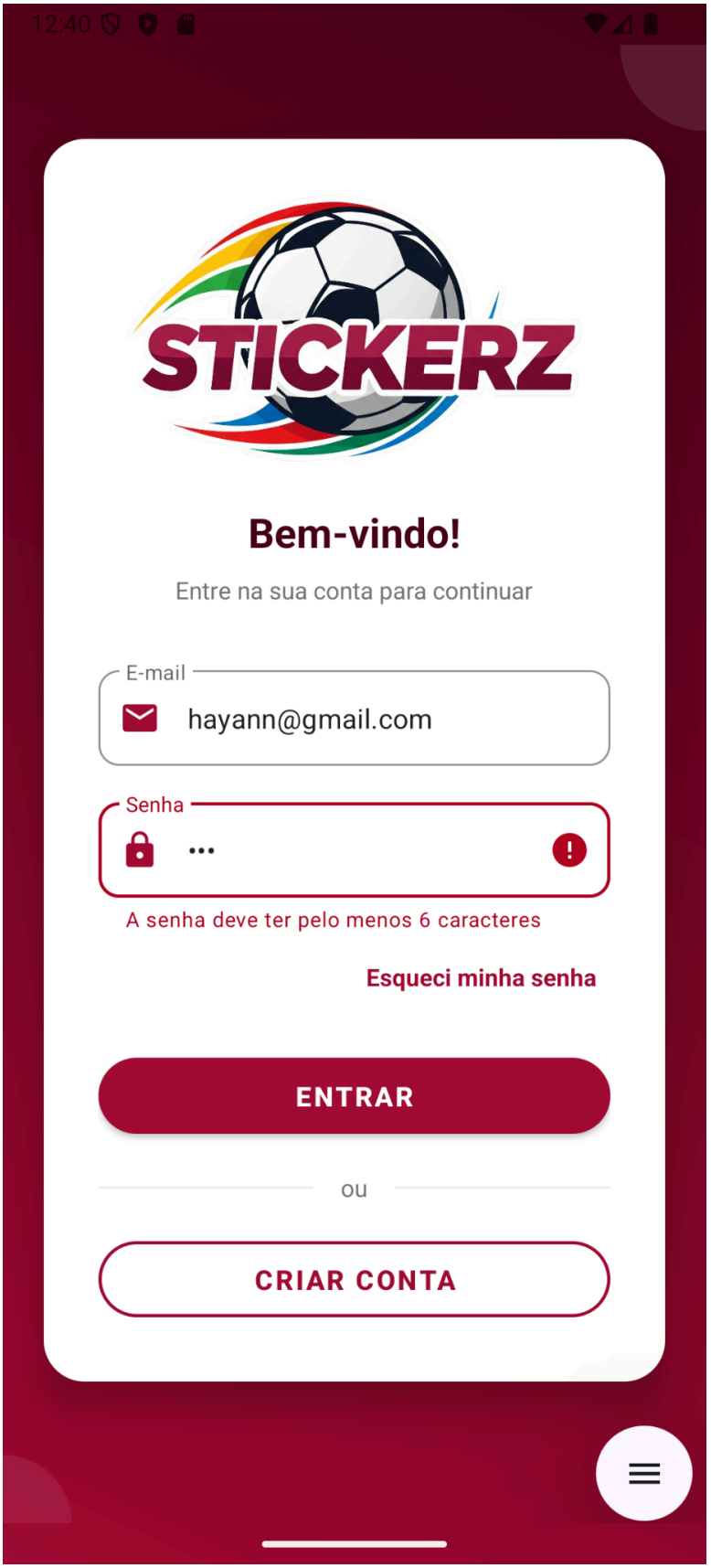
Após realizar o login o usuário será redirecionado para a tela inicial de explorar colecionadores, vista na figura 9.

Figura 5 - Tela de login.



Fonte: Autoria própria.

Figura 6 - Tela de login com validação.



Fonte: Autoria própria.

5.2 Tela de cadastro

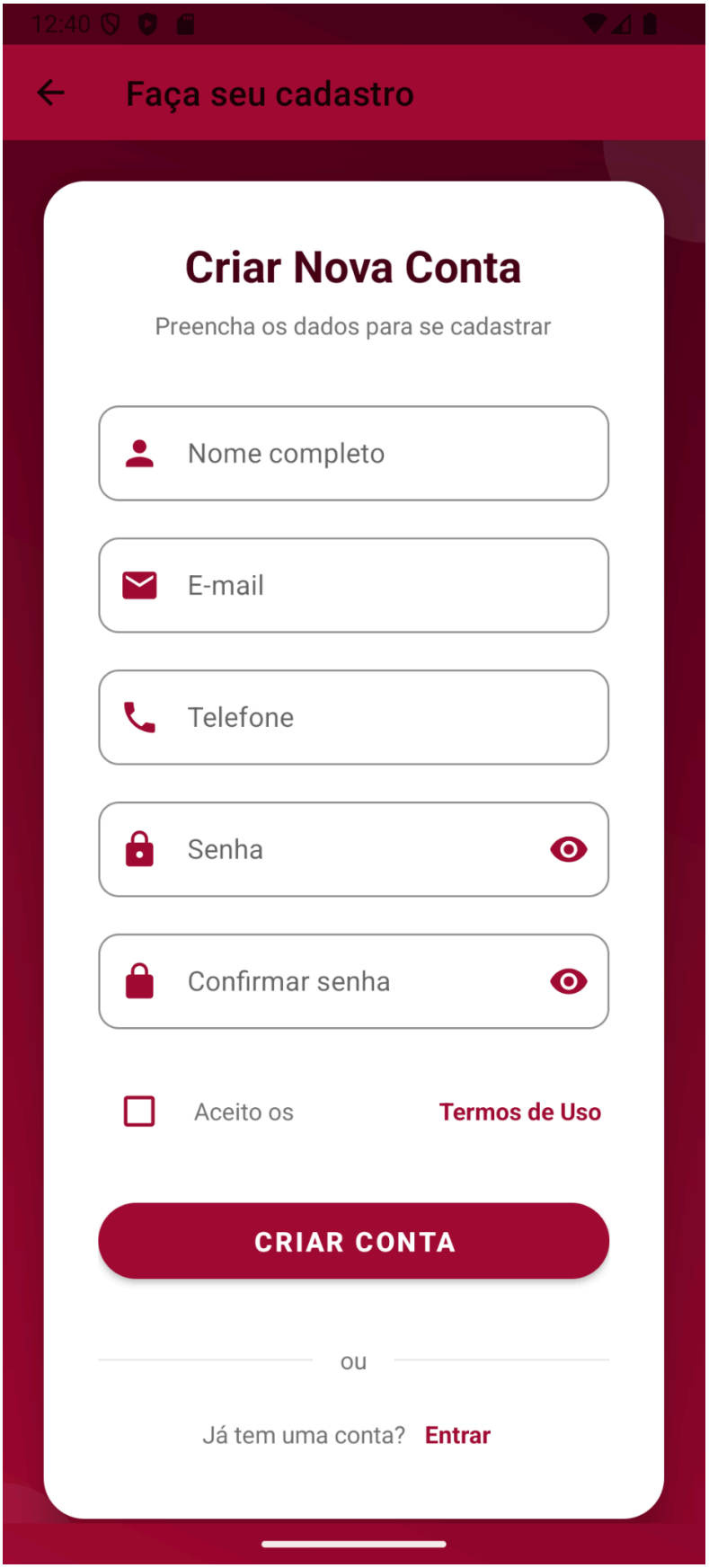
Na figura 7 vemos a tela de cadastro do aplicativo Stickerz, ela é acessada quando o usuário escolhe a opção de criar uma nova conta. Ela serve para coletar as informações necessárias para o registro.

O usuário deve preencher os campos de nome completo, e-mail, telefone, senha e confirmação de senha. Também é preciso aceitar os termos de uso antes de finalizar o cadastro. No final da tela, há um botão para criar a conta e um *link* para quem já possui conta, facilitando o retorno para a tela de *login*.

Caso o usuário digite alguma informação inconsistente, ou que não passe na validação de e-mail ou senha, o campo inválido ficará em destaque na cor vermelha e será exibida uma mensagem com o erro abaixo do campo como vemos na figura 8.

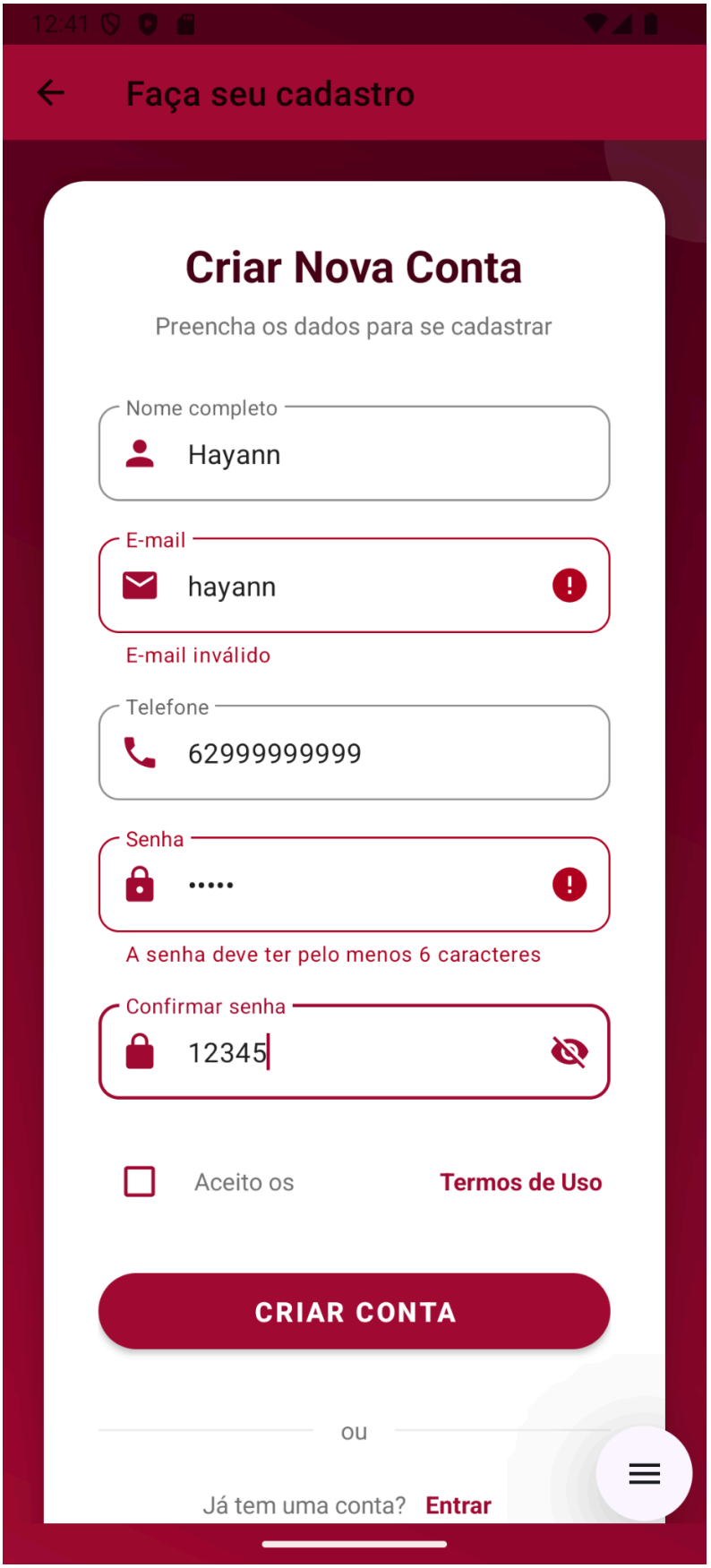
Após realizar o cadastro o usuário será redirecionado para a tela inicial de explorar colecionadores, vista na figura 9.

Figura 7 - Tela de cadastro.



Fonte: Autoria própria.

Figura 8 - Tela de cadastro com validação.



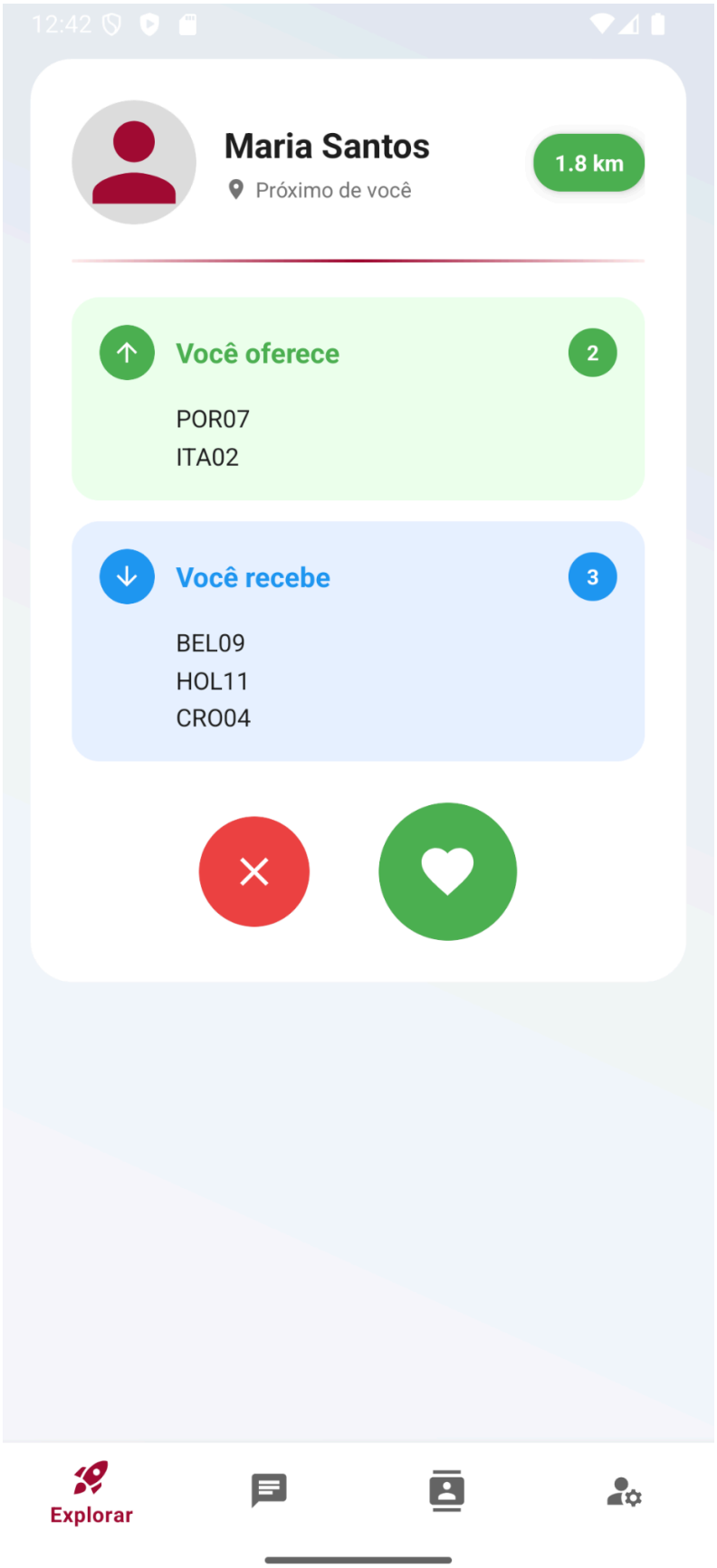
Fonte: Autoria própria.

5.3 Tela de encontrar colecionadores

Esta tela exibida na figura 9 é a de sugestões de trocas. Ela mostra o perfil de outro usuário próximo, com nome, distância e as figurinhas que podem ser trocadas. No campo “Você oferece” aparecem as figurinhas que o próprio usuário pode enviar. No campo “Você recebe” estão as figurinhas que o outro colecionador pode oferecer. Então o usuário tem a opção de demonstrar interesse clicando no botão verde ou então caso o usuário não tenha interesse na troca ele pode clicar no botão vermelho.

Abaixo, está o menu de navegação, onde é possível acessar outras partes do aplicativo que serão exibidas.

Figura 9 - Tela inicial, explorar



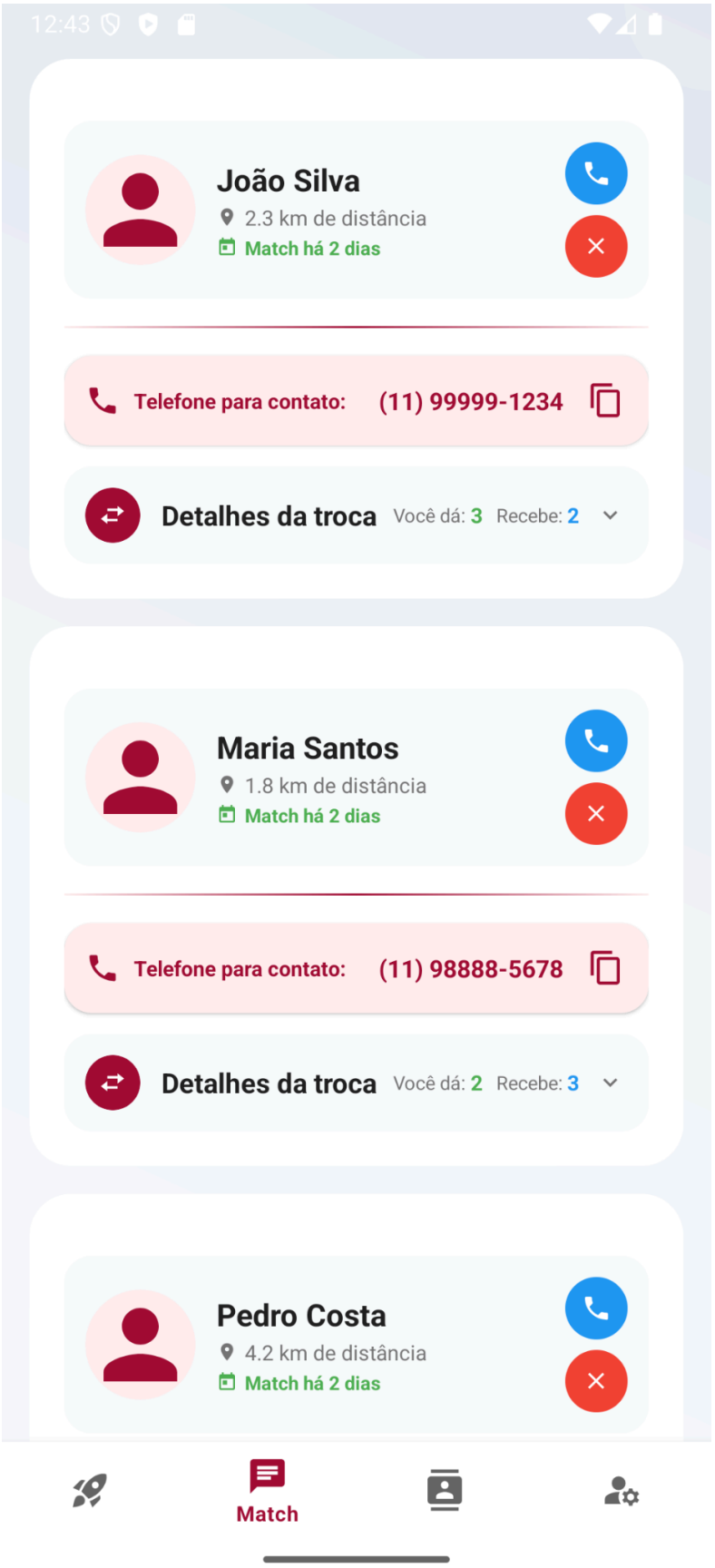
Fonte: Autoria própria.

5.4 Tela de matchs, exibição dos colecionadores compatíveis

Nesta tela, exibida na figura 10, o nome do colecionador aparece junto com a distância em quilômetros, facilitando a identificação de quem está mais próximo. Ao lado tem dois botões, o primeiro com símbolo de telefone abre uma tela perguntando se o usuário deseja ligar para o colecionador como mostra na figura 12 e o segundo botão vermelho com símbolo “x” serve para excluir o match com aquele colecionador em específico, ao clicar abre uma tela pedindo confirmação como visto na figura 13.

Mais abaixo, o telefone do colecionador é exibido em destaque, junto com um botão que permite copiar o número com facilidade. A seguir, está a seção "Detalhes da troca", que mostra de forma resumida a quantidade de figurinhas que o colecionador irá oferecer e receber. Quando essa seção é expandida, aparecem duas áreas bem definidas como visto na figura 11: a primeira com fundo verde claro, indicando as figurinhas que o colecionador oferece, e a segunda com fundo azul claro, apresentando as figurinhas que o colecionador vai receber.

Figura 10 - Tela de matchs, exibição de colecionadores compatíveis.

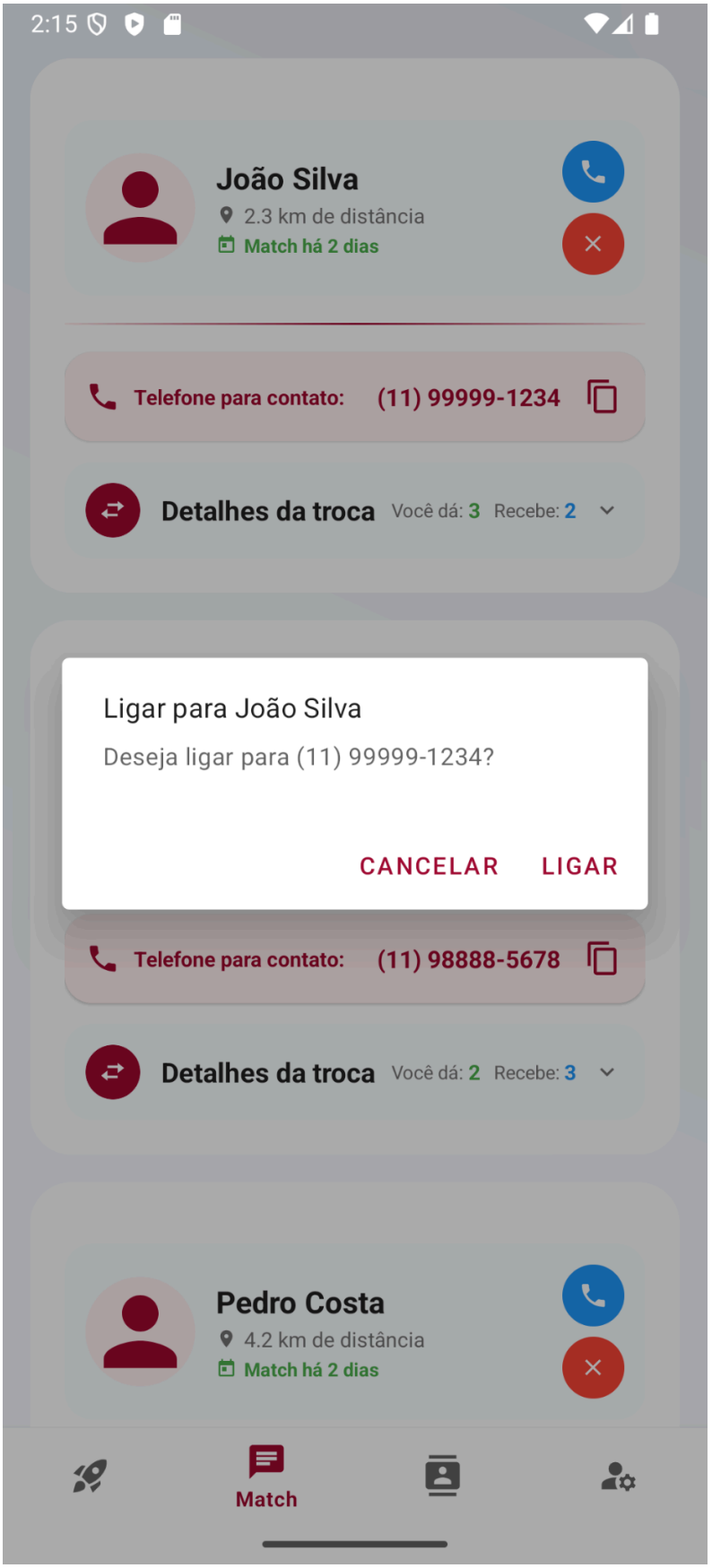


Fonte: Autoria própria.

Figura 11 - Tela de matchs, expandido.

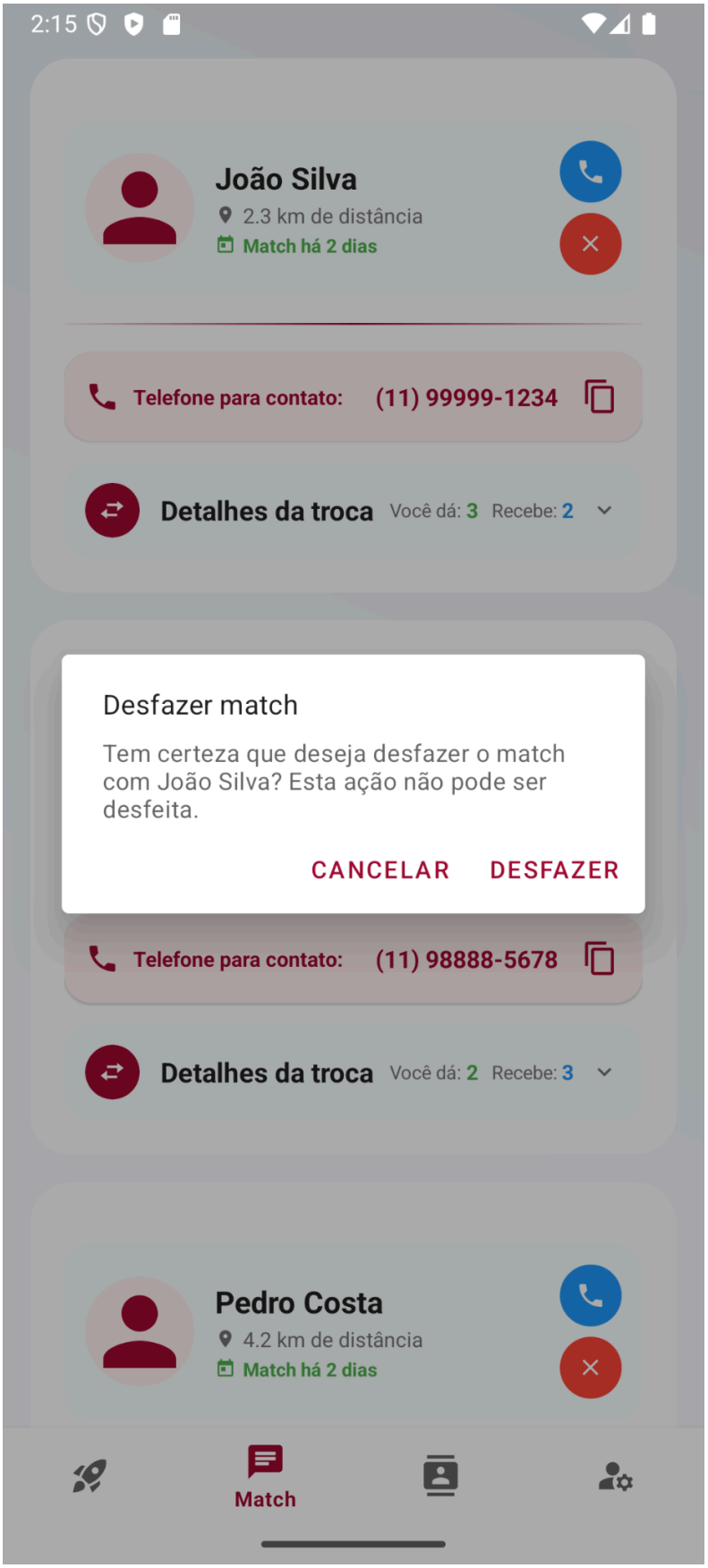
Fonte: Autoria própria.

Figura 12 - Tela de matchs, opção de ligar para um colecionador.



Fonte: Autoria própria.

Figura 13 - Tela de matchs, opção de excluir um colecionador



Fonte: Autoria própria.

5.5 Tela de álbum da copa

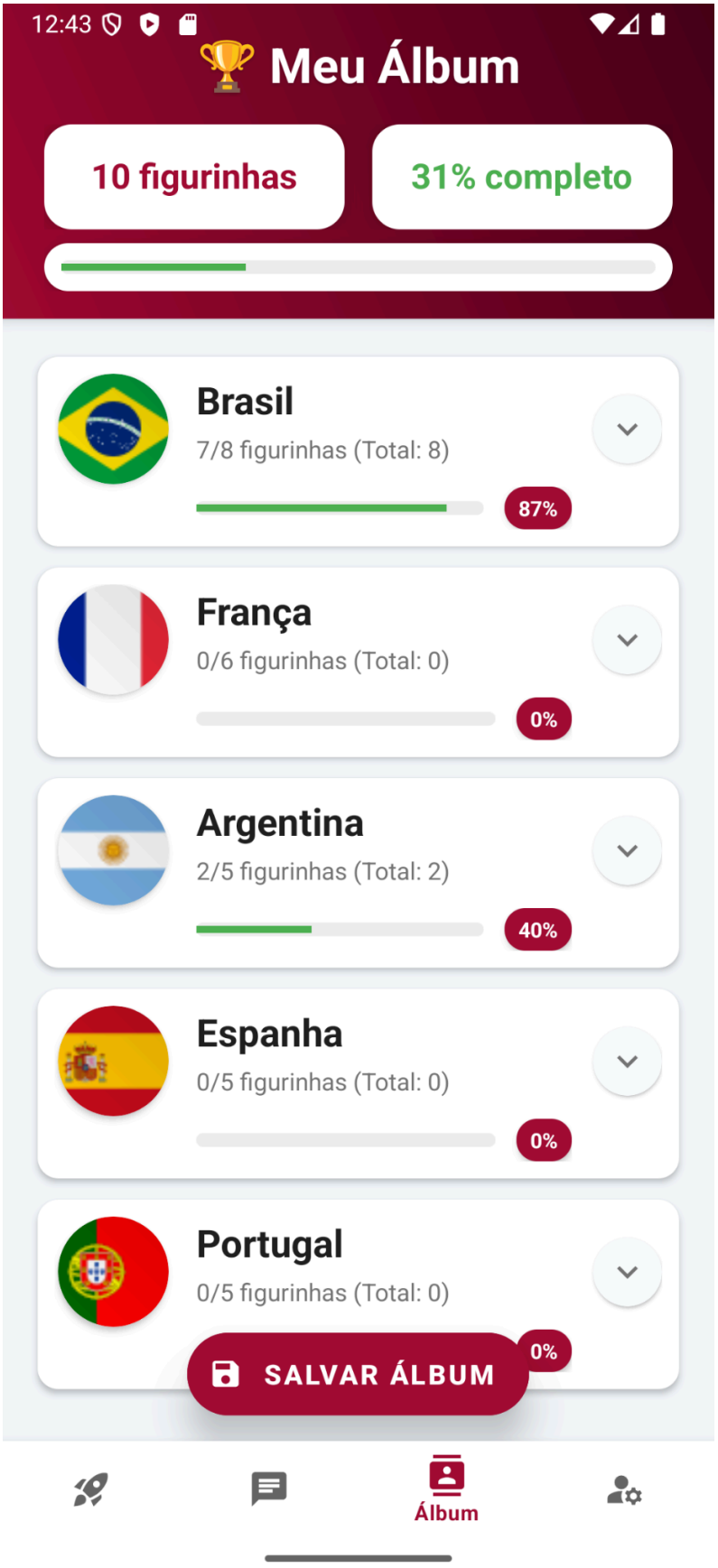
Na figura 14 vemos a tela de álbum, onde o colecionador vê o resumo e o progresso das figurinhas coletadas. No topo, o total de figurinhas coletadas aparece em vermelho e a porcentagem de conclusão aparece em verde. Logo abaixo, há uma barra de progresso que mostra visualmente o quanto o álbum está completo.

Cada país aparece em um *card* com sua bandeira, nome e o número de figurinhas coletadas sobre o total. A barra colorida indica o progresso daquele país e ao lado um botão que permite expandir ou recolher as figurinhas. Quando expandido, como vemos na figura 15, o colecionador vê a lista das figurinhas daquele país, com nome, status de coleta e dois botões: um vermelho com “-” para diminuir a quantidade e um verde com “+” para adicionar figurinhas.

Ao final, há um botão grande e vermelho escrito “salvar álbum” que o colecionador deve usar para guardar as alterações feitas.

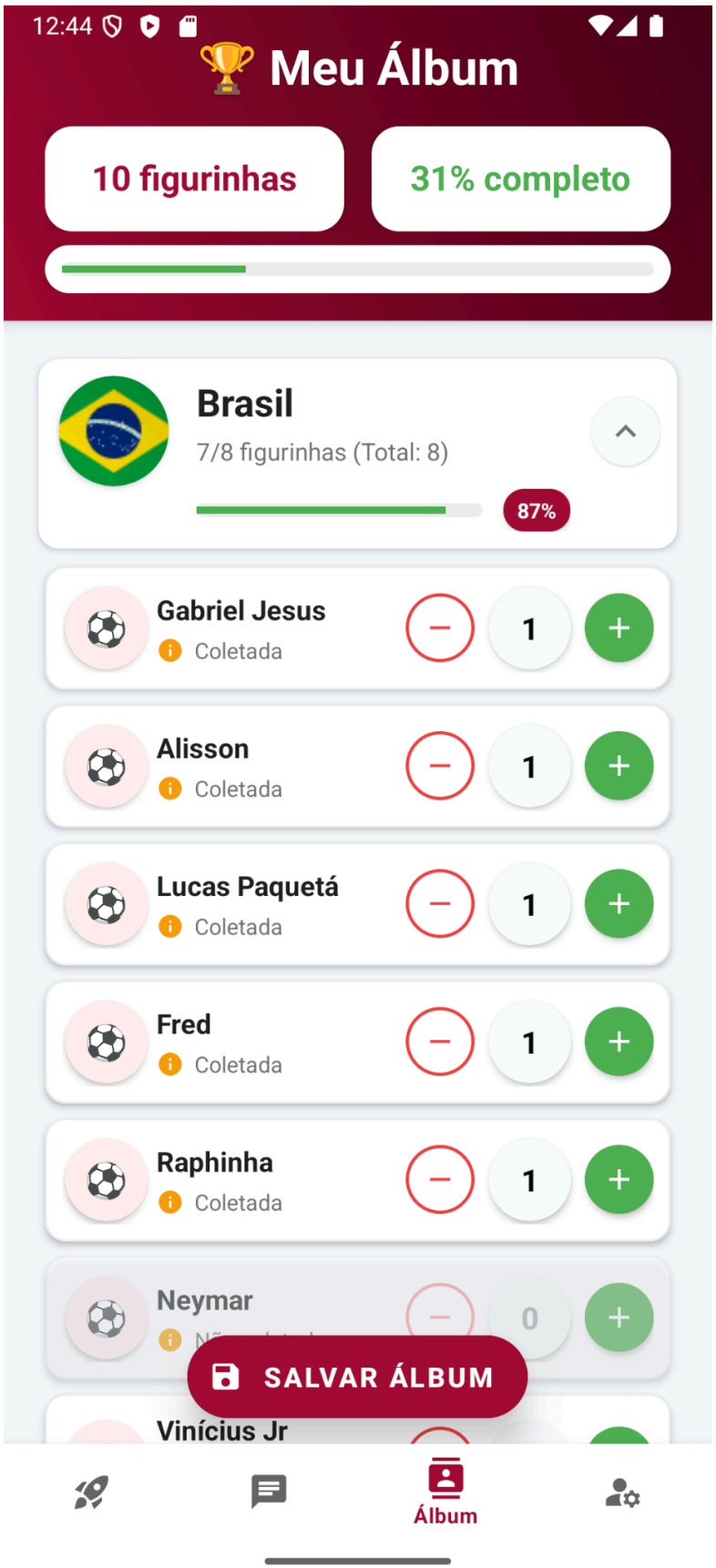
Essa tela busca os dados do servidor backend, da tabela TEAM_DATA e STICKERZ_DATA, como descrito no modelo conceitual no capítulo 4.4 e essas tabelas tão pré carregadas com os dados de cada time e cada figurinha, como descrito no capítulo 4.6.

Figura 14 - Tela de álbum da copa



Fonte: Autoria própria.

Figura 15 - Tela de álbum da copa, expandido.



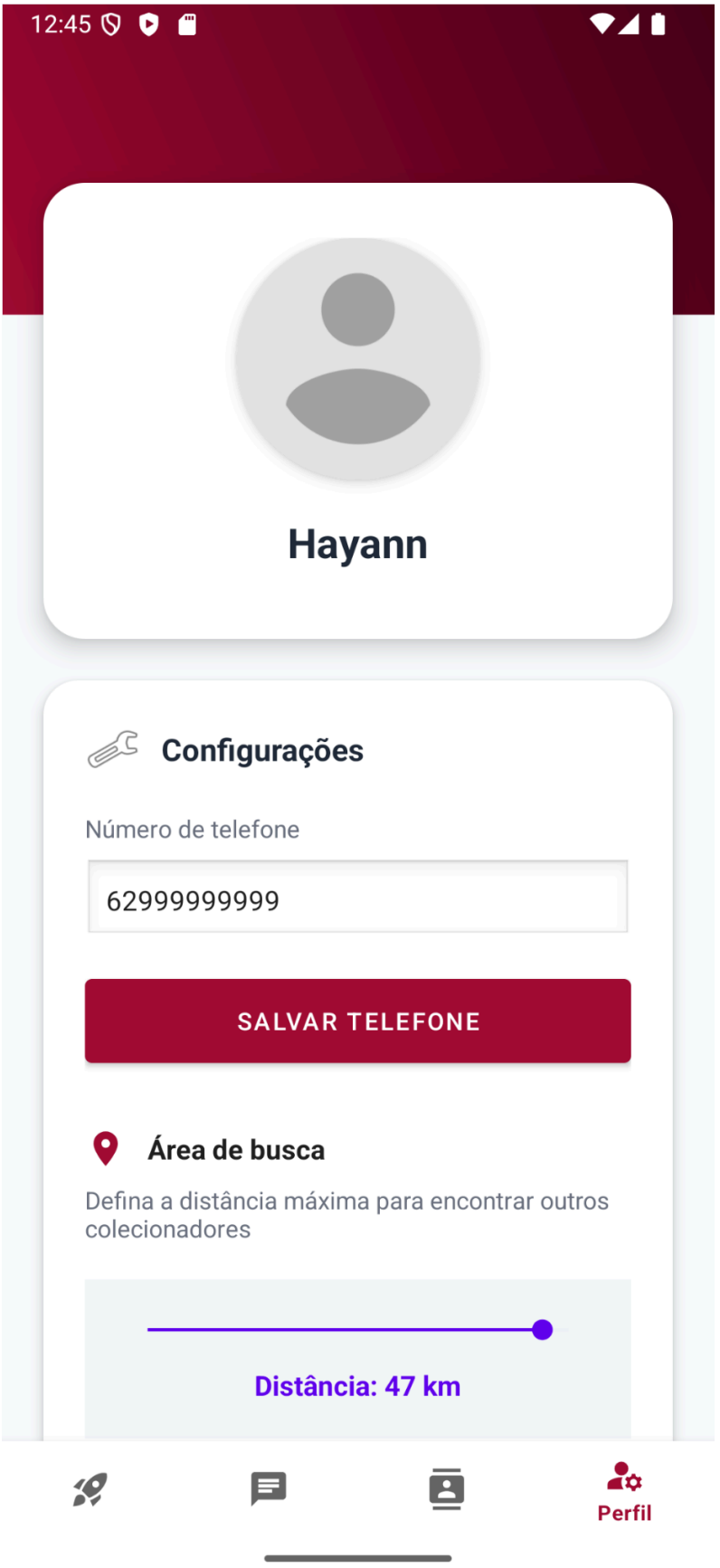
Fonte: Autoria própria.

5.6 Tela de perfil

Na figura 16 é a tela de perfil do usuário, com a foto e o nome destacados na parte superior. Logo abaixo, há campos onde o colecionador pode ajustar suas informações, como o telefone e a área máxima de busca.

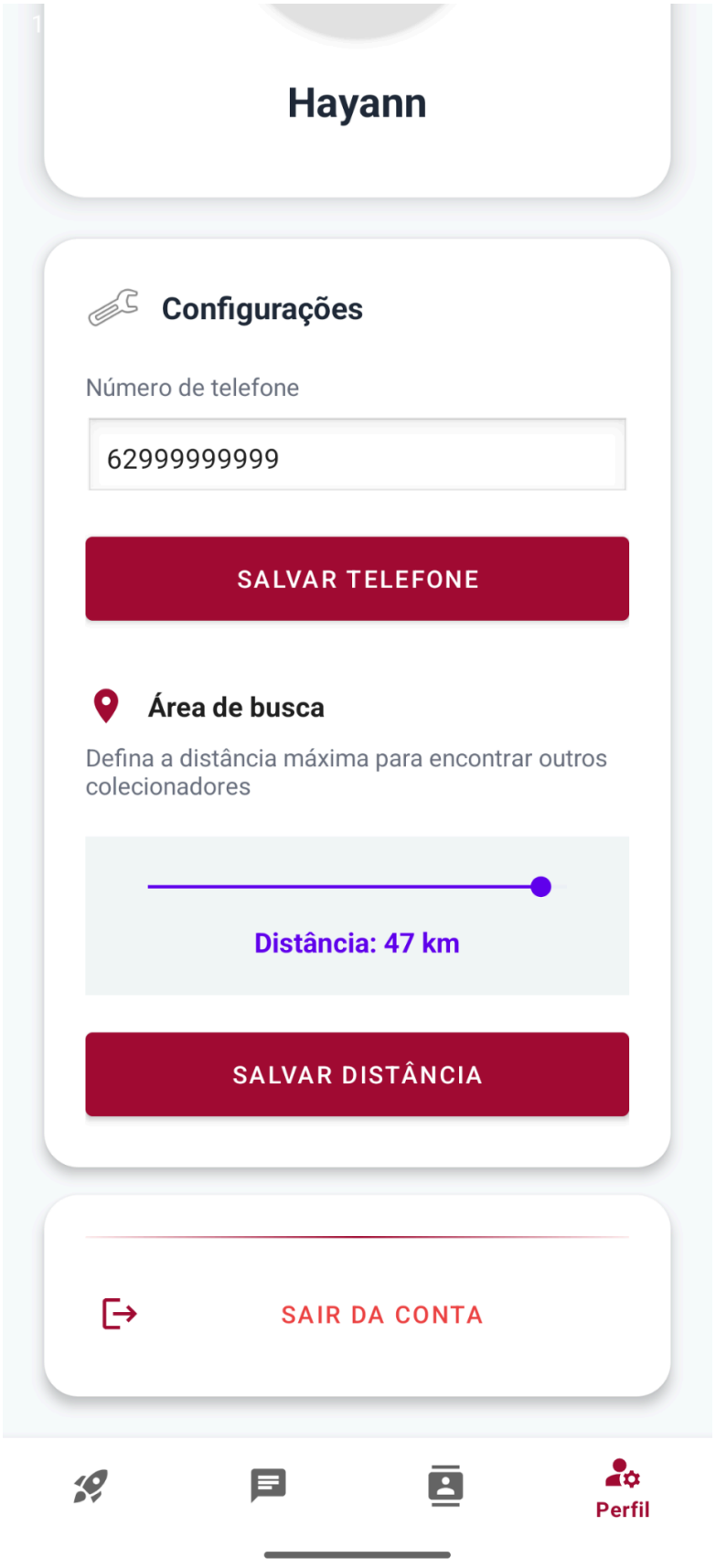
Quando quiser confirmar as alterações, o usuário toca nos botões de salvar. No final da tela tem o botão de sair da conta, ao tocar, surge um alerta pedindo confirmação como vemos na figura 18.

Figura 16 - Tela perfil.



Fonte: Autoria própria.

Figura 17 - Tela de perfil, continuação.



Fonte: Autoria própria.

Figura 18 - Tela de perfil, confirmação para sair.

Fonte: Autoria própria.

6 CONSIDERAÇÕES FINAIS

O desenvolvimento do aplicativo Stickerz representou uma oportunidade valiosa de aplicar na prática os conhecimentos técnicos adquiridos durante o curso de Ciência da Computação. O projeto conseguiu atingir seus objetivos principais ao criar uma solução digital funcional que facilita a troca de figurinhas da Copa do Mundo entre colecionadores.

A implementação do sistema permitiu a aplicação direta de conceitos estudados no curso. A programação orientada a objetos foi utilizada tanto no desenvolvimento Android com Kotlin quanto no backend com Kotlin e Spring Boot, aplicando princípios como encapsulamento, herança e polimorfismo. Os conhecimentos de banco de dados foram aplicados na modelagem conceitual e relacional, definindo entidades, relacionamentos e implementando a persistência através do Spring Data JPA.

A arquitetura de software foi um aspecto central do projeto, implementando o padrão MVVM no frontend mobile para separar adequadamente a lógica de apresentação dos dados, e o padrão MVC com camada de serviços no backend, garantindo a separação de responsabilidades conforme estudado nas disciplinas de engenharia de software.

O desenvolvimento de APIs REST consolidou os conhecimentos sobre protocolos HTTP, métodos de requisição e estruturação de endpoints, enquanto a implementação do sistema de autenticação JWT aplicou conceitos de segurança da informação estudados no curso. A utilização de tecnologias como Spring Security demonstrou a aplicação prática de frameworks para autenticação e autorização.

O uso de ferramentas como Android Studio, IntelliJ IDEA, DBeaver e Postman auxiliou o aprendizado sobre ambientes de desenvolvimento integrado e ferramentas de teste de APIs.

Como limitações identificadas, o projeto focou especificamente nas figurinhas da Copa do Mundo e a validação foi realizada através de testes funcionais, sem avaliação extensiva com usuários reais.

Para trabalhos futuros, sugere-se a expansão para outros tipos de colecionáveis, implementação de uma funcionalidade de chat para comunicação em tempo real e desenvolvimento de algoritmos mais sofisticados para otimização das sugestões de troca.

O projeto demonstrou que os conhecimentos adquiridos no curso são suficientes para desenvolver soluções tecnológicas completas e funcionais, usando na prática o aprendizado através da aplicação em um contexto real de desenvolvimento de software.

REFERÊNCIAS

BARCELOS, P. E. L.; LIMA, T. V.; DE AGUIAR, A. C. **Blogs e redes sociais na atenção à saúde da família: o que a comunicação online traz de novo?**. Revista Eletrônica de Comunicação, Informação & Inovação em Saúde, [S. l.], v. 14, n. 1, 2020. DOI: 10.29397/reciis.v14i1.1747. Disponível em: <https://www.reciis.icict.fiocruz.br/index.php/reciis/article/view/1747>. Acesso em: 14 abril 2025.

BRITANNICA. **Java computer programming language**. Disponível em: <https://www.britannica.com/technology/Java-computer-programming-language>. Acesso em: 20 maio 2025.

DEITEL, P. J.; **Android for programmers: An app-driven approach.** 2. ed. Philadelphia, PA: Prentice Hall, 2013.

DEVELOPER.ANDROID. **Visão geral do Kotlin**. Disponível em: <https://developer.android.com/kotlin/overview?hl=pt-br>. Acesso em: 20 maio 2025.

DEVELOPER.ANDROID. **Android Studio**. Disponível em: <https://developer.android.com/studio>. Acesso em: 10 maio 2025.

DBEAVER. ***Universal Database Tool***. Disponível em: <https://dbeaver.io/>. Acesso em: 20 maio 2025.

FIRMINO, Anderson. **Figurinha 'impossível' de Neymar no álbum da Copa chega a valer mais de sete salários mínimos; entenda**. 2022. Disponível em: <https://g1.globo.com/sp/santos-regiao/noticia/2022/08/22/figurinha-impossivel-de-neymar-no-album-da-copa-e-vendida-por-ate-r-9-mil-entenda.ghtml>. Acesso em: 20 nov. 2023.

GARGENTA, M. **Learning android: Develop mobile apps using java and eclipse.** 2. ed. Sebastopol, CA: O'Reilly Media, 2014.

GeeksforGeeks. **Difference Between MVC, MVP and MVVM Architecture Pattern in Android.** Disponível em: <https://www.geeksforgeeks.org/difference-between-mvc-mvp-and-mvvm-architecture-pattern-in-android/>. Acesso em: 11 maio 2025.

GeeksforGeeks. **MVVM (Model View ViewModel) Architecture Pattern in Android.** Disponível em: <https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/>. Acesso em: 11 maio 2025.

JETBRAINS. **O IDE para desenvolvedores profissionais**. Disponível em: <https://www.jetbrains.com/pt-br/idea/>. Acesso em: 20 maio 2025.

KOTLINLANG. **Visão geral do Kotlin**. Disponível em: <https://kotlinlang.org/>. Acesso em: 10 maio 2025.

MICROSOFT. **Model-View-ViewModel (MVVM)**. Disponível em: <https://learn.microsoft.com/pt-br/dotnet/architecture/maui/mvvm>. Acesso em: 20 abril 2025.

MYSQL. **Why MySQL?**. Disponível em: <https://www.mysql.com/why-mysql/>. Acesso em: 20 abril 2025.

POSTMAN. **Your Complete API Platform, From Design to Delivery**. Disponível em: <https://www.postman.com/>. Acesso em: 20 maio 2025.

REDHAT. **What is a REST API?** Disponível em: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>. Acesso em: 10 abril 2025.

ROSA, G. M. **O Problema do Colecionador de Cupons: quanto custa completar o álbum de figurinhas da Copa do Mundo?** Disponível em: <https://proceedings.sbmac.emnuvens.com.br/sbmac/article/view/3617>. Acesso em: 21 maio 2025.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O estudante Hayann Gonçalves Silva do Curso de Ciência da Computação, matrícula 20192002800460, telefone: 62 985244291, e-mail hayanngs@hotmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado "Desenvolvimento do Software 'STICKERZ': Aplicativo Para Troca de Figurinhas da Copa do Mundo", gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 28 de Março de 2025.

gov.br Documento assinado digitalmente
HAYANN GONCALVES SILVA
Data: 28/03/2025 18:15:55-0300
Verifique em https://validar.iti.gov.br

Assinatura do autor: _____

Nome completo do autor: _____

Assinatura do professor–orientador: _____ gov.br Documento assinado digitalmente
LUCILIA GOMES RIBEIRO
Data: 28/03/2025 23:15:43-0300
Verifique em https://validar.iti.gov.br

Nome completo do professor-orientador: _____